SDVietNam

We bring your dream to life

YÊU CẦU PROTOCOL CHO SDALARM VERSION 2.0

General date: 20.11.2015

Updated date: 30.12.2015

Edition 2nd: 10.01.2016

Edition 3nd: 09.05.2016

Edition 4nd: 11.06.2016

Edition 5nd: 13.03.2017

→ edited: rtcswitch thành sdalarm, bổ sung chức năng delete task, update theo logic mới

1. Ứng dụng:

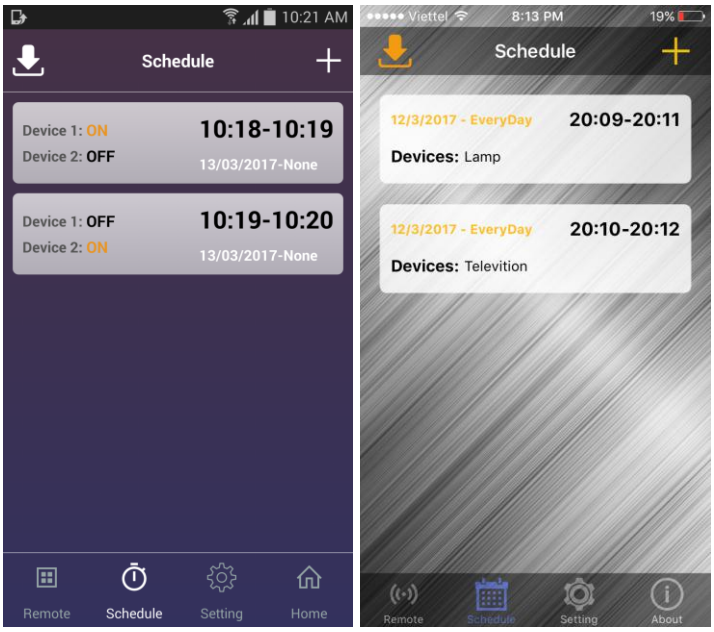
Các dòng sản phẩm của Smart Switch nói chung đều phục vụ mục tiêu IOT theo định hướng của nhóm.

Với sản phẩm smart switch người dùng có thể xây dựng bất cứ gói dữ liệu nào mong muốn và truyền đạt cho các thiết bị để bắt chúng phải làm việc.

Với smart switch phiên bản sdalarm người dùng chỉ cần dùng smart device của mình lên lịch trình đóng ngắt các thiết bị của mình theo các ngày trong 1 tháng với độ chính xác giờ, phút.

Ứng dụng của sdalarm có thể cho các smart farm, nông dân có thể lên lịch tưới tiêu cho sản phẩm của mình,... hoặc có thể dùng trong các ngôi nhà thông minh để lên lịch đóng ngắt thiết bị điện trong nhà khi bạn đi công tác hay du lịch.

2. Giao diện đề xuất:



Các phần chính cần có trong schedule:
1 schedule sẽ có nhiều task ứng với mỗi ngày trong tháng, lưu ý: vì bộ nhớ phần cứng có giới hạn nên có thể version đầu tiên sẽ giới hạn người dùng chỉ được 1 task trong 1 ngày và lên lịch trong 1 tháng.
Như vậy trong 1 tháng sẽ có tối đa là 31 task.

- Trong giao diện chính của schedule:
  + Danh sách các task đã được tạo.
  + Các task có thể chỉnh sửa được.
  + Có thể thêm hoặc xóa các task.

- Trong 1 task sẽ có các nội dung sau:
  + Tên thiết bị cần điều khiển: chẳng hạn lamp, fan, tivi,...
  + Thời gian bắt đầu: chỉ nhận dữ liệu ngày, tháng và giờ phút.
  + Trạng thái đèn muốn điều khiển (chỉ cần ON) lúc bắt đầu.
  + Thời gian kết thúc: chỉ nhận dữ liệu ngày, tháng và giờ phút.
  + Theo logic trạng thái đèn sẽ là OFF lúc kết thúc thời gian trên.

3. Khung dữ liệu:

Data frame cho software:

+ 1 byte request:

Byte[0] = 0x8B

Ý nghĩa:

	Start bit	**des bit**	**version bits**				**confirm bits**	
Bit	7	6	5	4	3	2	1	0
Version 2	1	0	0	0	1	0	1	1

+ 3 bytes control cho Remote:

Byte	0	1	2
Value	Start	State 1	State 2

Ý nghĩa:

- **Start byte = 0xA1**

	Start bit	**des bit**	**Request bits**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	0	1	0	0	0	0	1

- **State bytes =0x84 or 0x85**

	Start bit	**des bit**	**Version bits**	**State bit**	**Means**

Bit	7	6	5	4	3	2	1	0	
Version 2	1	0	0	0	0	1	0	0	**off**
Version 2	1	0	0	0	0	1	0	1	**on**

+ 1 byte Reset phần cứng cho Remote:

Ý nghĩa:

- **Start byte = 0xA2**

	Start bit	**des bit**	**Request bits**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	0	1	0	0	0	1	0

+ 14bytes schedule data:

		Begin time					**End time**					**Device**	
Byte	0	1	2	3	4	5	6	7	8	9	10	11	12
Value	Sta	Da	Mo	Ye	Ho	Mi	Da	Mo	Ye	Ho	Mi	St1	St2

Ý nghĩa:

- **Start byte = 0xA3**

	Start bit	**des bit**	**Request bit**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	0	1	0	0	0	1	1

- **Day, Month, Hour, Year, Minute byte:**

Trong đó:Minute <= 59 (111011→ 6 bits), Year chỉ lấy 2 số cuối

	Start bit	**des bit**	**Data bit**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	0	x	x	x	x	x	x

Với **byte Day của Begin Time**:

	Start bit	**des bit**	**Ev bit**	**Data bit**				
Bit	7	6	5	4	3	2	1	0
Version 2	1	0	x	x	x	x	x	x

Ev bit = 0: tính lặp lại hàng ngày không được dùng đến. Khi đó data của End time vẫn được set bình thường.
Ev bit = 1: sử dụng chức năng lặp lại hằng ngày trong 1 tháng. Khi đó data của End time = 0x00

- **State byte = 0x84 or 0x85**

	Start bit	**des bit**	**Version bits**					**State bit**	**Means**
Bit	7	6	5	4	3	2	1	0	
Version 2	1	0	0	0	0	1	0	0	**off**
Version 2	1	0	0	0	0	1	0	1	**on**

Với Des bit: đích truyền từ software hoặc firmware
0: từ soft → firm
1: từ firm → soft

+ 7 bytes cài đặt thời gian cho system data:

		System Time					
Byte	0	1	2	3	4	5	6
Value	Start	Da	Mo	Ye	Ho	Mi	Se

Ý nghĩa:

- **Start byte = 0xA4**
- **System Time:** dùng để cài đặt thời gian cho hệ thống phần cứng, thời gian này được application lấy tự động từ hệ thống, sau khi reset thành công, firmware sẽ gửi lên 1 gói data để confirm (xem phần firmware)

Ví dụ:

Ở tab Remote: trạng thái của các device sẽ được điều khiển như sau:
Byte[0] = 0xA1: start byte cho frame 1
Byte[1] = 0x84 hoặc 0x85
Byte[2] = 0x84 hoặc 0x85

Ở tab schedule:
Ví dụ 1: dữ liệu sẽ gửi xuống firmware sau khi application đã tạo task và lên lịch **bắt đầu ngày 20/11/2015 vào lúc 10h59 và kết thúc ngày 21/11/2015 vào lúc 10h59 sẽ on device 1** như sau:

Byte[0] = 0xA3: start byte cho frame 2
Byte[1] = 0x94: Ngày 20
Byte[2] = 0x8B: Tháng 11
Byte[3] = 0x8F: Năm 15
Byte[4] = 0x8A: 10 giờ
Byte[5] = 0xBB: 59 phút

Byte[6] = 0x91: Ngày 21
Byte[7] = 0x8B: Tháng 11
Byte[8] = 0x8F: Năm 15
Byte[9] = 0x8A: 10 giờ
Byte[10] = 0xBB: 59 phút
Byte[11] = 0x85: device 1 on
Byte[12] = 0x84: device 2 off → theo logic mới device 2 don't care trong trhop này

Ví dụ 2: dữ liệu sẽ gửi xuống firmware sau khi application đã tạo task và lên lịch **bắt đầu ngày 20/11/2015 vào lúc 10h59 và kết thúc ngày 21/11/2015 vào lúc 10h59 sẽ on device 2** như sau:

Byte[0] = 0xA3: start byte cho frame 2
Byte[1] = 0x94: Ngày 20
Byte[2] = 0x8B: Tháng 11
Byte[3] = 0x8F: Năm 15
Byte[4] = 0x8A: 10 giờ
Byte[5] = 0xBB: 59 phút
Byte[6] = 0x91: Ngày 21
Byte[7] = 0x8B: Tháng 11
Byte[8] = 0x8F: Năm 15
Byte[9] = 0x8A: 10 giờ
Byte[10] = 0xBB: 59 phút
Byte[11] = 0x84: device 1 off → theo logic mới device 2 don't care trong trhop này
Byte[12] = 0x85: device 2 on

Ví dụ 3: dữ liệu sẽ gửi xuống firmware sau khi application đã tạo task và lên lịch **bắt đầu ngày 20/11/2015 vào lúc 10h59 và kết thúc ngày 21/11/2015 vào lúc 10h59 sẽ on cả 2 device** như sau:

Byte[0] = 0xA3: start byte cho frame 2
Byte[1] = 0x94: Ngày 20
Byte[2] = 0x8B: Tháng 11
Byte[3] = 0x8F: Năm 15
Byte[4] = 0x8A: 10 giờ
Byte[5] = 0xBB: 59 phút
Byte[6] = 0x91: Ngày 21
Byte[7] = 0x8B: Tháng 11
Byte[8] = 0x8F: Năm 15
Byte[9] = 0x8A: 10 giờ
Byte[10] = 0xBB: 59 phút
Byte[11] = 0x85: device 1 on
Byte[12] = 0x85: device 2 on

Như vậy logic mới sẽ có những điểm cần lưu ý sau:
+ Có thể hẹn giờ riêng cho từng thiết bị.
+ Trạng thái điều khiển trong suốt thời gian xảy ra là ON cho mỗi hoặc cả 2 device.
+ Trạng thái điều khiển sau khoảng thời gian của task sẽ là OFF.
+ Vẫn có thể xảy ra 2 task cùng lúc nếu trên 2 device khác nhau.
+ Không quan tâm trạng thái hiện tại của tab remote.

Với chức năng lặp lại hàng ngày everyday chẳng hạn: bắt đầu lúc 10h59 và kết thúc lúc 0h59 sẽ on device 2 và lặp lại hằng ngày bắt đầu từ ngày 20/11/2015 thì dữ liệu như sau:

Byte[0] = 0xA3: start byte cho frame 2
Byte[1] = 0xB4: bắt đầu từ ngày 20 lặp lại hằng ngày
Byte[2] = 0x8B: Tháng 11
Byte[3] = 0x8F: Năm 15
Byte[4] = 0x8A: 10 giờ
Byte[5] = 0xBB: 59 phút
Byte[6] = 0x00
Byte[7] = 0x00
Byte[8] = 0x00
Byte[9] = 0x80: 0 giờ
Byte[10] = 0xBB: 59 phút
Byte[11] = 0x84: device 1 off → theo logic mới don't care trhop này.
Byte[12] = 0x85: device 2 on

Với chức năng delete task: chức năng này sẽ được delete theo từng task trên app, mỗi lần delete app sẽ tự transfer data của các task còn lại trong list, nếu là task cuối cùng, app sẽ gửi các mã 0x00 như sau:

Byte[0] = 0xA3: start byte cho frame 2
Byte[1] = 0x00
Byte[2] = 0x00
Byte[3] = 0x00
Byte[4] = 0x00
Byte[5] = 0x00
Byte[6] = 0x00
Byte[7] = 0x00
Byte[8] = 0x00
Byte[9] = 0x00
Byte[10] = 0x00
Byte[11] = 0x00: device 1 don't care
Byte[12] = 0x00: device 2 don't care

➔ **Fw** sẽ sử dụng các mã này để erase bộ nhớ đã lưu data của các task trước đây.

➔ Sau khi đã xóa hết tất cả các task trong list **Sw** sẽ **không cho phép** sử dụng chức năng **transfer** thêm lần nào nữa.

➔ Số lượng task giới hạn hiện tại phụ thuộc vào bộ nhớ của chip là **10** task, do đó **Sw** và **Fw** cần có cơ chế ngăn user tạo quá số lượng task này.

+ 21 bytes thiết lập lại SSID và pass wifi:

		Data					
Byte	0	1	...	...	...	...	20
Value	Start	Byte1	Byte2	Byte	Byte	Byte	Byte20

Ý nghĩa:

- **Start byte: A5**
- **Byte 1 → Byte 10:** các ký tự để tạo nên SSID, tối đa 10 ký tự.
- **Byte 11 → Byte 20:** các ký tự để tạo nên pass wifi, tối đa 10 ký tự.

Quy định chung:

Đối với SSID: tối thiểu 4 ký tự dạng chữ (a,b,c....)

Đối với pass: tối thiểu 8 ký tự dạng số hoặc chữ (a,b,c,1,2,3....)

Firmware sẽ thực hiện việc kiểm tra số lượng ký tự này và phản hồi **Invalid** hay **Valid**

Trường hợp SSID hoặc pass chưa đủ số lượng 10bytes thì các byte còn lại sẽ là **0xFE**

Ví dụ:

User muốn đặt ssid là abcd, với pass là 12345678, thì các byte gửi xuống như sau:

A561 62 63 64 fe fe fe fe fe fe31 32 33 34 35 36 37 38 fe fe

Trường hợp các byte **0x00 và 0xFF** sẽ không được tính là 1 ký tự, các ký tự phải được quy định theo bảng mã ASCII vì vậy sẽ không có các ký tự đặc biệt và ký tự NULL

Data frame cho firmware:

+ 3 bytes confirm: gửi lúc nhận được 1 byte request từ application

- **Start byte = 0xCB**

Ý nghĩa:

	Start bit	**des bit**	**version bits**				**confirm bits**	
Bit	7	6	5	4	3	2	1	0
Version 2	1	1	0	0	1	0	1	1

- **State bytes =0xC4 or 0xC5**

	Start bit	**des bit**	**Version bits**					**State bit**	**Means**
Bit	7	6	5	4	3	2	1	0	

Version 2	1	1	0	0	0	1	0	0	**off**
Version 2	1	1	0	0	0	1	0	1	**on**

Với Des bit: đích truyền từ software hoặc firmware
0: từ soft → firm
1: từ firm → soft

+ 3 bytes confirm cho Remote tab: gửi lúc nhận được 3 bytes control từ application

- **Start byte = 0xE1**

	Start bit	**des bit**	**Request bits**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	1	1	0	0	0	0	1

- **State bytes = 0xC4 or 0xC5**

	Start bit	**des bit**	**Version bits**					**State bit**	**Means**
Bit	7	6	5	4	3	2	1	0	
Version 2	1	1	0	0	0	1	0	0	**off**
Version 2	1	1	0	0	0	1	0	1	**on**

+ 1 byte confirm cho Remote tab: gửi lúc nhận được 1 byte yêu cầu reset phần cứng
Ý nghĩa: byte[0] = 0xE2

+ 3 bytes confirm cho Schedule tab: gửi mỗi lần nhận được task mới từ tab Schedule

- **Start byte = 0xE3**

	Start bit	**des bit**	**Request bit**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	1	1	0	0	0	1	1

- **2 Data confirm bytes: Byte[1], Byte[2]**

	Start bit	**des bit**	**confirm data bits**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	1	x	x	x	x	x	x

Trong đó:
Byte[1] Byte[2]: Xác nhận dữ liệu nhận được từ **begin vàEnd: valid or invalid**
Byte[1] = byte[2] = 0xC0: **invalid**
Byte[1] = byte[2] = 0xC1: **valid**
Valid: là tất cả các byte từ software gửi xuống đều được kiểm tra và hợp lý.
Invalid: là tất cả các byte từ software gửi xuống mà có bất kỳ 1 byte nào sai.

Trong trường hợp firmware xác nhận chưa nhận được bất cứ dữ liệu nào từ schedule thì software cần truyền lại gói tin khác đến khi nào nhận được đúng confirm từ firmware.

+ 2 bytes xác nhận việc cài đặt thời gian hệ thống:

- **Start byte = 0xE4**
- **2 data confirm byte: byte[1] byte[2]**

	Start bit	**des bit**	**confirm data bits**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	1	x	x	x	x	x	x

Ý nghĩa các bit để xác định các giá trị **valid** hay **invalid** tương tự như trên.

Ví dụ:

+ Khi nhận được byte request từ software, firmware gửi lên như sau:
Byte[0] = 0xCB: start byte cho frame 1
Byte[1] = 0xC4 hoặc 0xC5: state of device 1 hiện tại
Byte[2] = 0xC4 hoặc 0xC5: state of device 2 hiện tại

+ Khi nhận được control byte từ software, firmware gửi lên như sau:
Byte[0] = 0xE2: start byte cho frame 2
Byte[1] = 0xC4 hoặc 0xC5: state of device 1 hiện tại
Byte[2] = 0xC4 hoặc 0xC5: state of device 2 hiện tại

+ Khi nhận được gói 7 setting timer database từ software, firmware gửi lên như sau:
Byte[0] = 0xE3: start byte cho frame 3
Byte[1] = Byte[2] = 0xC1: nếu tất cả database đều nhận được valid
Chẳng hạn có 1 byte Day **không nhận được hoặc nhận được mà không hợp lệ**:
Byte[1] = Byte[2] = 0xC0

+ 3 bytes xác nhận ssid và pass:

- **Start byte: 0xE5**
- **2 bytes xác nhận:**

	Start bit	**des bit**	**confirm data bits**					
Bit	7	6	5	4	3	2	1	0
Version 2	1	1	x	x	x	x	x	x

Ý nghĩa:

2 bytes xác nhận: **0xC0: Invalid và 0xC1: Valid**
Tương tự như các byte xác nhận cho thời gian nhưng khác ở chỗ: byte[1] xác nhận tính hợp lý cho ssid và byte[2] xác nhận tính hợp lý cho pass

Ví dụ:
+ Sau khi kiểm tra nếu thấy **ssid và pass đều hợp lý** thì firmware sẽ gửi lên: **E5 C1 C1**
+ Sau khi kiểm tra nếu **ssid sai** mà **pass đúng** thì firmware gửi lên: **E5 C0 C1**
+ sau khin kiểm tra nếu **ssid đúng** mà **pass sai** thì firmware gửi lên: **E5 C1 C0**

4. **Thực hiện:**
   Software deveopler: Trương Minh Giới
   Firmware developer: Nguyễn Văn Vui, Lê Chí Bình
   Hardware: Phúc Lê